An Anonymization Service for Software-Defined Networks

Leonardo H. S. Bomfim
Instituto Federal de Sergipe - Campus Propriá
Propriá, Sergipe, Brasil
Email: leonardo.bomfim@ifs.edu.br

Edilayne M. Salgueiro
Universidade Federal de Sergipe
São Cristóvão, Sergipe, Brasil
Email: edilayne@ufs.br

Ricardo J. P. de B. Salgueiro
Universidade Federal de Sergipe
São Cristóvão, Sergipe, Brasil
Email: salgueiro@ufs.br

***Resumo*—This paper presents a service to make the anonymization in SDN to guarantee the availability of the services in the network through the concealment of the stations. Because of this it was developed the anonymizer BomIP using C Language and Libpcap Library, then it was included as a service on RunOS OpenFlow Controller. To validate this service it were made three cases studies on a simulated environment of a Denial of Service attack. The results shows that BomIP has a running time 65% better than others anonymizers, besides BomIP guarantees that all packets can be tracked and a mitigation of 80% from the attacks trials, supporting the services provides by the newtork to continue running.**

***Index Terms*—SDN, Security, Controller.**

I. Introdução

As Redes Definidas por Software (SDN) consistem na ideia da separação do plano de dados do plano de controle da rede [1]. O que resulta em transformar a rotina de dispositivos da rede tradicional, em simples *switches* cujo trabalho corresponde apenas a encaminhar a política que é designada por um controlador centralizado e programável. Assim, SDN representa uma das mais novas tecnologias de arquitetura de redes devido a dois motivos [2]: primeiro por permitir a separação entre a seção de dados e o controle da rede, permitindo que o controle da rede não seja mais em nível de *hardware*, e sim, por *software*. Outro motivo, implementa de forma inteligente e centralizada toda a informação da rede, facilitando tomada de decisões e soluções para serviços de segurança e proteção aos sistemas.

Uma técnica para fornecer segurança em uma SDN consiste em utilizar um serviço de anonimização para realizar o ocultamento de serviços e estações [3]. Este ocultamento acontece através da anonimização dos endereços IP da rede, impedindo que um usuário, ao invadir o ambiente, tenha acesso aos endereços reais de serviços e estações disponíveis, impedindo desta forma um ataque ou obtenção de informações trafegadas na rede.

Anonimização consiste em remover ou modificar todos os campos de identificação (nome, identidade, etc.) de um usuário ou entidade, de forma a não permitir a identificação do mesmo, garantindo assim a segurança da informação [4]. Logo, o objetivo da anonimização é proteger a privacidade dos usuários, não permitindo que informações sejam disponibilizadas de forma pública permitindo a identificação.

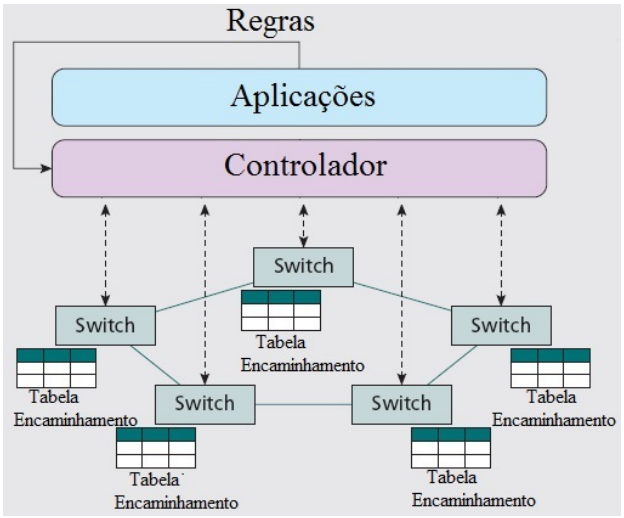
O mesmo ocorre relacionado aos dados dos pacotes de tráfego de uma rede, a anonimização pode ser realizada em campos que identificam uma rede e a localização de serviços e estações, como endereços MAC, endereços IP e portas. Neste ponto, o controlador SDN tem um papel importante para que um serviço de anonimização seja utilizado [3], de forma que ele possa realizar a anonimização dos dados dos pacotes trafegados na rede, e realize o roteamento dos mesmos de forma correta convertendo os dados de tráfego anonimizados nos valores reais no momento de rotear na rede, garantindo assim, que os pacotes não sejam perdidos.

No cenário atual para empresas, em que informação é um recurso estratégico, o que acarreta em um alto custo para coletar, manter e gerenciar estes dados [5], desenvolver técnicas para prover segurança para essas informações, de forma que elas fiquem disponíveis e não possam ser invadidas por usuários maliciosos torna-se de fundamental importância. Com base neste contexto, este trabalho apresenta um serviço de anonimização para Redes Definidas por Software com o objetivo de mitigar as tentativas de ataque de Negação de Serviço sofridas por uma rede, o que pode acarretar em problemas como comprometimento das informações e indisponibilidade dos serviços. Os resultados obtidos demonstram que é possível reduzir a taxa de ataques em relação à uma rede semelhante que não utiliza o serviço desenvolvido.

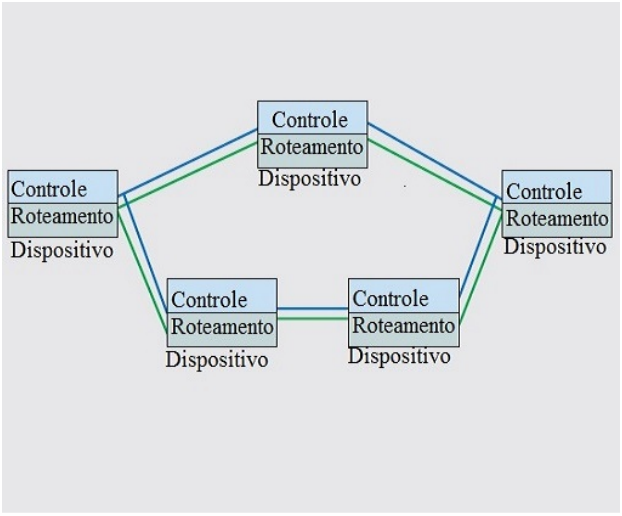
A Figura 1(a) corresponde a arquitetura SDN, em que os dispositivos da rede correspondem a *switches* que seguem as regras aplicadas pelo controlador central, enquanto que na Figura 1(b) é apresentada a arquitetura tradicional de uma rede, em que cada dispositivo é responsável por realizar as funções de controle e encaminhamento de pacotes.

O plano de controle é responsável por monitorar a rede, tomar as decisões de roteamento e realizar a programação do comportamento físico da rede. O plano de dados é formado por *switches* que são conectados de forma a corresponderem a rede física. A separação entre o plano de dados e o controlador SDN torna-se uma arquitetura fundamental para as redes futuras; no entanto, também cria desafios à segurança. Do ponto de vista de segurança, a preocupação inicial foi do controle centralizado como um ponto de falha na rede [6]. Se todas as decisões são realizadas em um local e este é invadido, permitirá o controle da rede pelo usuário invasor.

Serviços de segurança em uma SDN são importantes para



(a) Arquitetura SDN



(b) Arquitetura de rede Tradicional

Figura 1. Exemplo de Arquitetura SDN (a) e Tradicional (b)

impedir que a rede possa sofrer, por exemplo, um ataque de Negação de Serviço, levando a comprometer os serviços que são disponibilizados pela rede [7]. Pacotes do tipo *spoof*, usados em diferentes tipos de mensagens pelo protocolo OpenFlow (*Handshake*, *Hello* etc.) e lógica mal elaborada no controlador podem ser fatores que levem a este tipo de ataque.

Serviços voltados para segurança em SDN estão em constante pesquisa e desenvolvimento [1], para impedir que a identidade dos serviços disponibilizados na rede possam ser descobertas e tornem-se alvos de ataques de usuários maliciosos. Diversos serviços encontram-se em desenvolvimento, dentre as técnicas abordadas no momento uma de destaque consiste em anonimizar os endereços IP gerenciados pelo controlador [3], de forma a ocultar as estações e servidores, reduzindo riscos de ataques de *spoof* e de Negação de Serviço.

Este trabalho está dividido da seguinte forma: a Seção II aborda os conceitos sobre Redes Definidas por Software, enquanto que a Seção III explica sobre Anonimização. A Seção IV apresenta a Arquitetura Proposta. A Seção V traz Trabalhos Relacionados referente à pesquisas que estão sendo desenvolvidas sobre segurança em SDN com anonimização e a Seção VI traz os Estudos de Caso e resultados obtidos. E na Seção VII são apresentadas as conclusões e trabalhos futuros.

II. Redes Definidas por Software

A demanda por serviços de redes tem crescido rapidamente, e desta forma, dispositivos de rede com tráfego de diferentes fontes, como de vídeos, *big data* e dispositivos móveis tornam-se cada vez mais um desafio para os operadores de redes [8]. Além da maior quantidade de servidores e máquinas virtuais aumentando o tráfego entre servidores.

A grande maioria dos dispositivos de redes tradicionais concentram as funcionalidades de controle e fluxo dos dados [9]. Porém, os administradores de redes necessitam configurar e gerenciar cada um dos dispositivos separadamente. Para superar este desafio, os administradores necessitam de uma rede eficiente, flexível, ágil e escalonável. Com base nestes problemas, surge o conceito de Redes Definidas por Software (SDN - *Software-Defined Network*) que refere-se a uma nova abordagem para tornar redes programáveis, com capacidade para iniciar, controlar, alterar e gerenciar o comportamento da rede dinamicamente através de uma interface [10]. A proposta de redes programáveis alavancam a pesquisa de métodos de gerenciamento e configuração de redes automatizados.

O conceito consiste em uma arquitetura de rede emergente onde o controle da rede é dissociado e separado do mecanismo de envio, e é diretamente programável. Desta forma, em SDN há um controlador central lógico que possui a rede e controla múltiplos encaminhamentos de pacotes entre dispositivos (por exemplo: *switches*) configurados via interfaces.

Em uma arquitetura SDN existem três partes distintas separadas por camadas: aplicação, controlador e plano de dados. A camada de aplicação indica a parte que explora a dissociação do controle e do plano de dados para obter metas específicas, como mecanismos de segurança ou soluções de gerenciamento de internet. Está é a camada em que aplicações e serviços definem o comportamento da rede.

As aplicações comunicam-se com o controlador usando a interface *northbound* (Figura 2). O plano de controle é a parte que manipula o encaminhamento dos dispositivos pelo controlador para atingir o objetivo específico. O controlador utiliza a interface *southbound* da SDN para os *switches* se conectarem ao plano de dados.

O plano de dados é responsável por lidar com os pacotes baseados nas instruções recebidas pelo controlador [10]. Normalmente, o plano de dados é o ponto final dos serviços do controlador e aplicações, não sendo apenas responsável por encaminhar, descartar e alterar pacotes, ele também possui recursos para realizar a classificação dos pacotes.

O controlador corresponde a uma abstração completa da infra-estrutura de rede, permitindo ao administrador aplicar políticas e protocolos customizados pela rede física. A interface *northbound* representa a camada entre as aplicações do

controlador e a plataforma SDN. O plano de dados representa o encaminhamento do *hardware* na rede de arquitetura SDN. Devido ao controlador precisar comunicar-se com a infraestrutura de rede, certos protocolos são necessários para realizar este controle e gerenciamento, sendo que a interface chamada *southbound*, mais conhecida, corresponde ao OpenFlow.

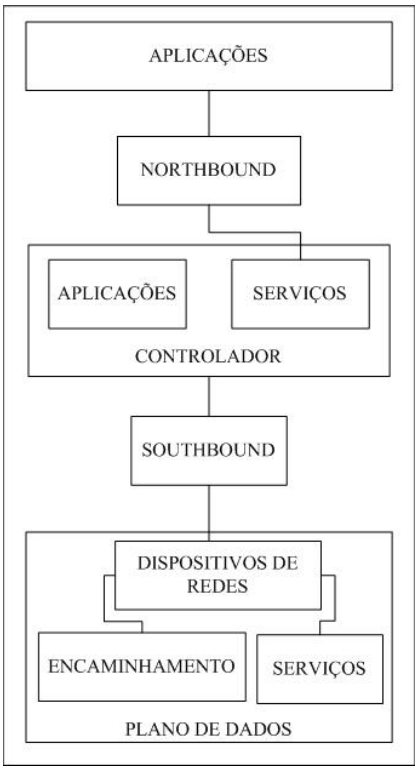


Figura 2. Componentes de um SDN
Fonte: [1]

No paradigma tradicional de redes, centrado no *hardware*, *switches* são sistemas fechados que possuem controlador, plano de dados e suporte específico de acordo com o fabricante. Assim, novos protocolos e serviços costumam ser um desafio, e os *switches* precisam ser atualizados. Em contraste com SDN, em que os *switches*, podem ser reprogramados para suportar novas tecnologias de comunicação.

A. Protocolo OpenFlow

OpenFlow é um protocolo usado para gerenciamento da interface *southbound* da arquitetura SDN. Corresponde a primeira interface padronizada para facilitar a interação entre o controlador e o plano de dados, provendo um acesso por meio de *software* para a tabela de fluxos com instruções para os *switches* e os roteadores sobre como direcionar o tráfego.

Assim, o protocolo OpenFlow define uma API de comunicação entre o controlador e o *switch*. No entanto, tanto o controlador quanto o *switch* devem implementar o protocolo OpenFlow. O controlador manipula o fluxo de tabelas do *switch* pela adição, atualização e deleção de entrada de fluxos. O *switch* OpenFlow suporta o tráfego do fluxo através da manutenção de uma ou mais tabelas de fluxos.

Um tráfego pode ser definido como um particionamento de fluxos, onde cada fluxo pode ser, por exemplo, uma conexão TCP, em que pacotes com os mesmos endereços IP ou MAC ou VLAN, ou pacotes que chegam de uma mesma porta. A arquitetura do OpenFlow consiste, basicamente, de uma quantidade numerosa de *switches* ou outros equipamentos OpenFlow, que são gerenciamentos por controladores OpenFlow.

Cada registro na tabela de fluxo possui três campos [11]: (i) *Header*: que especifica informações como origem e destino, ID, portas, endereços IP e Ethernet. (ii) Ação: que especifica como o pacote deve ser processado (encaminhado para alguma porta ou perdido ou enviado ao controlador). (iii) Estatística: que inclui informação como número de pacotes, *bytes* e tempo entre pacotes.

Quando um *switch* recebe um pacote que possui uma perda na tabela de encaminhamentos, este pacote pode ser encaminhado ao controlador, passado para a próxima tabela de fluxos ou descartado. Se o pacote for encaminhado para o controlador, o mesmo decide como tratar o pacote, podendo descartar o pacote ou adicionar a um fluxo, o que indica ao *switch* a direção de como encaminhar um pacote semelhante no futuro.

Um *switch* OpenFlow possui uma ou mais tabelas de fluxos que realizam *lookups* e encaminhamentos. O controlador OpenFlow gerencia os *switches* através de um canal com conexão segura, em que este canal corresponde a interface que conecta cada *switch* OpenFlow do plano de Dados com o controlador.

B. Vantagens de SDN

Conforme mencionado, SDN apresenta vantagens em relação à arquitetura padrão de rede centrada no *hardware*. A separação do controlador e do plano de dados apresenta diversas vantagens, conforme indicado em [8]:

- Tornou-se fácil programar aplicações devido as abstrações providas pela plataforma do controlador e as linguagens de programação em redes podem ser compartilhadas;
- Todas aplicações podem ser beneficiadas por terem a mesma informação da rede (visão global da rede), levando a uma maior consistência e eficácia de decisões de políticas;
- Aplicações podem tomar ações (por exemplo, reconfigurar encaminhamento de dispositivos) em qualquer parte da rede. Assim, não há necessidade de elaborar uma estratégia sobre a localização da nova funcionalidade;
- A integração de diferentes aplicações torna-se mais fácil interfaces programáveis. Por exemplo, balanceamento de carga e roteamento podem ser combinados sequencialmente, com decisões sobre balanceamento sendo tomadas a partir de rotinas de policiamento.

C. Problemas e Questões Abertas

SDN possui questões pendentes relacionadas à segurança, principalmente por permitir que desenvolvimentos sejam realizados sobre a arquitetura, o que pode permitir que diversos ataques sejam realizados na rede, como sugerem alguns autores. Uma análise geral de segurança em SDN é apresentada em [12], que concluiu que o controlador sendo centralizado e a rede programável permite que novas ameaças sejam introduzidas.

De acordo com [13] há três grupos de ameaças voltados para SDN: baseado no funcionamento dos componentes de SDN, na natureza e tipo de ataque e baseado nos tipos de recursos. Por exemplo, um ataque de *Spoofing* utiliza recursos do sistema como ARP, MAC e IP para forjar o endereço da fonte do ataque, realizando ataques como Negação de Serviço. Outro ataque é o *Tampering*, que refere-se a uma intencional e não autorizada destruição ou modificação dos componentes da rede, como topologia, lista de acesso, tabela de fluxos e políticas.

D. Segurança em SDN

Um aspecto importante em SDN consiste na segurança, para permitir a disponibilidade de interações entre a rede e aplicações para um eficiente controle de acesso, e garantir a proteção dos recursos da rede contra qualquer tipo de ataque. Em particular, as políticas de segurança para SDN devem garantir que os recursos sejam devidamente protegidos contra ações que coloquem em risco a rede ou as aplicações.

Os serviços providos devem ter procedimentos para assegurar a confiabilidade dos *softwares* embutidos em nós da rede. Tais procedimentos devem incluir o comportamento dos componentes do *software*, detectar vulnerabilidades etc. Estas medidas de segurança devem ser ativadas durante a inicialização da SDN, e também em atividades que implicam em atualizações de sistemas. Contudo, procedimentos de segurança não são especificados em SDN, o que se torna um desafio prático existente para operacionalização da rede.

Uma análise geral de segurança em SDN é apresentada em [12]. Nesse trabalho o autor concluiu que o controlador sendo centralizado e a rede programável permite que novas ameaças sejam introduzidas. Um grande número de análises tem sido realizadas indicando que, alterações nos elementos e nos relacionamentos no *framework* SDN, introduziram novas vulnerabilidades, que não estavam presentes antes da SDN.

Sistemas de monitoramento são essenciais para proteger redes de ataques. Em [14] foi apresentado um método para detectar ataques de Negação de Serviço Distribuído (DDoS), utilizando controlador OpenFlow baseado em C++ (NOX). Outra abordagem é a utilizada pelo OpenSAFE [15], que faz uso da linguagem especificação de fluxo ALARMS (*A Language for Arbitrary Route Management for Security*). Segundo os autores essa linguagem simplifica o gerenciamento do monitoramento da rede.

SDN têm duas propriedades que se tornaram atrativas para usuários maliciosos. Primeiramente, o controle da rede encontrar-se realizado por *software*, que sempre está sujeito a falhas e a possuir vulnerabilidades. E segundo, a centralização do controlador, que permite que quaisquer usuários que tenham acesso a este componente, poder controlar a rede completamente.

Em [12] são descritas potenciais ameaças para redes SDN:

1) Vulnerabilidade dos *switches*: de forma que apenas um *switch* pode ser usado para realizar perdas ou desvio de pacotes numa rede ou mesmo para sobrecarregar o controlador.
2) Ataques ao controlador: que pode gerar ataque de Negação de Serviço ou para captura de informações.
3) Vulnerabilidades no controlador: que consiste em um dos mais perigosos. Uma falha ou um controlador malicioso pode comprometer toda a rede.
4) Falta de mecanismos para garantir segurança entre o controlador e o gerenciamento de aplicações: o que corresponde a uma ameaça semelhante a segunda, ou seja, as falhas de segurança suscetíveis aos *softwares*.
5) Vulnerabilidade de estações em ambientes administrativos: comum em redes tradicionais, a rede é atacada a partir de estações de trabalho em ambientes corporativos. A diferença é que em redes SDN, devido ao controlador ser centralizado, o acesso através de uma estação pode comprometer toda a rede.
6) Falta de confiança em recursos para análises forenses: o que permite compreender a causa de um problema detectado e realizar procedimentos mais rápidos para que a rede volte a operar normalmente.
7) Fluxos com tráfegos falsos: pode ser usado para atacar *switches* e controladores. Esta ameaça pode ser disparada por uma falha de dispositivos ou por um usuário malicioso. O ataque pode utilizar elementos da rede (servidores, computadores etc) para iniciar um ataque de Negação de Serviço contra o *switch* OpenFlow e recursos do controlador.

III. Anonimização

Em Redes de Computadores, a anonimização é a modificação dos dados de tráfego em uma rede para proteger e preservar as identidades dos pontos de origem e destino dos registros compartilhados. A anonimização tende a preservar propriedades de tráfego da rede para análise, mas de forma a não permitir que uma aplicação consiga rastrear a rede analisada [16], realizando o ocultamento dos serviços e estações de uma rede.

A anonimização tem por característica realizar o ocultamento de estações e serviços disponíveis em uma rede através da modificação dos dados do tráfego que identificam o endereço de destino do pacote. Este ocultamento tem o objetivo de impedir que um usuário malicioso, ao realizar uma análise do tráfego, consiga obter os endereços de serviços e estações existentes na rede. Desta forma, ele não realiza ataques como Negação de Serviço e Homem-no-Meio, por não ser capaz de obter o endereço real dos serviços.

As técnicas de anonimização de dados, em relação aos tipos de dados em um tráfego de redes, são classificadas em Macro-

Dados e Micro-Dados [17]. Na técnica de Macro-Dados são adicionados dados sintéticos para obter privacidade nos registros, de forma que todos os valores sejam modificados. Enquanto que na técnica de Micro-Dados trabalha-se a nível dos campos da rede, apenas os que contém dados de identificação (endereço MAC, IP, portas) do usuário são modificados. Estes dados são criptografados e podem ser utilizados para análise do tráfego da rede.

Uma técnica de Micro-Dados é a preservação de prefixo [18]. Esta técnica diz que dois endereços IP $a = a_1a_2...a_n$ e $b = b_1b_2...b_n$ possuem o mesmo $k - bit$ prefixo quando $0 \leq k \leq n$, se $a_1a_2...a_k = b_1b_2...b_k$ e $a_{k+1} \neq b_{k+1}$, indicando assim, que os endereços IP possuem o mesmo prefixo. Isto permite que, ao realizar a anonimização dos endereços IP em um *trace offline* ou em tempo real, novas sub-redes sejam criadas de forma anonimizadas, o que permite uma poster análise do tráfego da rede devido a manutenção da rastreabilidade, garantindo verificar a existência de possíveis ataques à rede, como ataque de Negação de Serviço dentre outros.

IV. Serviço de Anonimização BomIP

A arquitetura proposta consiste em um serviço de anonimização para uma Rede Definida por Software, com o propósito de realizar a mitigação da possibilidade de ataques à rede através da ocultação das estações. Este processo de ocultação das estações é realizado com a anonimização dos endereços IP da rede, inclusive da tabela mantida pelo controlador, e desta forma, impedindo que toda a rede seja descoberta em um ataque.

A. Anonimizador BomIP e Controlador RunOS

O anonimizador BomIP é uma classe desenvolvida neste trabalho utilizando a Linguagem de Programação C e a biblioteca Libpcap [19], que permite que seja utilizada como *plugin* em aplicações de análise de pacotes em uma rede, podendo também estes dados serem fornecidos através de um *trace*.

A Linguagem C foi escolhida para prover melhor desempenho durante o processamento dos dados recebidos na rede, para desta forma, minimizar possíveis perdas de pacotes durante o fluxo de dados. Para facilitar o acesso aos dados de um pacote, o anonimizador BomIP utiliza a biblioteca Libpcap [20].

Libpcap é uma biblioteca *opensource* que provê uma interface de alto nível para trabalhar com captura de pacotes em rede, tendo sido desenvolvida em 1994 na Universidade de Berkeley e posteriormente aprimorada [19]. O principal objetivo era criar uma API independente que eliminasse a necessidade de um sistema de captura. Esta biblioteca foi desenvolvida para ser trabalha com C e C++, no entanto, há mecanismos que permitem utilizá-la com *Perl*, *Python*, *Java*, *CSharp* e *Ruby*.

O anonimizador BomIP utiliza a técnica de preservação de prefixo através de um algoritmo de randomização dos valores gerados para substituir as informações reais. Por utilizar duas técnicas para atingir o objetivo de anonimizar a informação, o BomIP é considerado um anonimizador híbrido.

O processo de anonimização do BomIP é apresentado na Figura 3, este processo é realizado para os endereços IP de origem e destino, sendo análogo para ambos os tipos de endereço. Em um primeiro momento, o endereço IP tem os quatro octetos separados, porque o processo de anonimização é realizado separadamente para cada um dos octetos. Em seguida a anonimização é iniciada para o primeiro octeto x; assim, no vetor que armazena os valores anonimizados, consulta-se na posição x se já existe um valor gerado. Se não existir um valor, na posição x do vetor consta como um *flag* de valor -1 sinalizando um valor inexistente, e então um valor y é gerado randomicamente. Com este valor y é realizada uma consulta no vetor para verificar se o mesmo já existe, se o mesmo já existir, um novo valor randomicamente é gerado, e este processo é interrompido apenas quando um valor gerado ainda não existir no vetor. Quando um y não existente no vetor é gerado para posição x, este valor é retornado para formar o novo endereço IP anonimizado, e sempre que este valor x aparecer em qualquer octeto de qualquer endereço IP (origem ou destino), será substituído pelo valor y.

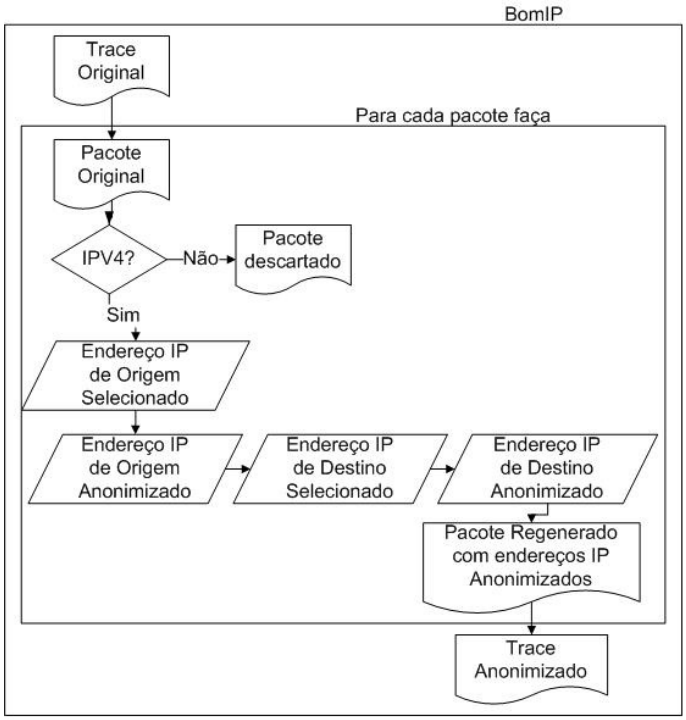


Figura 3. Processo de anonimização de endereços IP do BomIP

Importante observar que os valores são armazenados em um vetor, e como são armazenadas as possibilidades de valores em um octeto, o espaço de armazenamento corresponde a um vetor de 256 posições (valores de 0 a 255), isto porque não são armazenados todos os valores possíveis de IP, o que ocasiona um ganho de espaço na memória para o processamento. Este vetor é inicializado com todas as posições contendo o *flag* de valor -1, o que identifica que para uma dada

posição, ainda não existe um valor randomizado para ser utilizado na anonimização. A Figura 4 ilustra este processo, em que as posições 0 e 250 ainda não foram utilizadas para anonimização, ou seja, nenhum endereço IP, até o momento do processamento não conteve um octeto que fossem os valores 0 ou 250, enquanto que as demais posições já foram utilizadas e por isso possuem valores para anonimização. Por exemplo, sempre que um octeto tiver o valor 3, será anonimizado pelo valor 9.

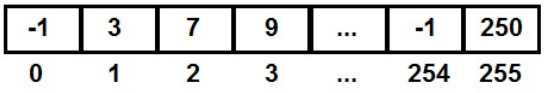


Figura 4. Exemplo de Vetor Anonimizado

B. Implementação do BomIP

RunOS [21] é um controlador OpenFlow desenvolvido em C++ em 2014 com o objetivo de ter uma alta performance, combinando técnicas de programação para atingir este objetivo, permitindo que o controlador tenha qualidade, programabilidade e usabilidade.

O controlador RunOS utiliza *threads* dedicadas para realizar a comunicação com os *switches* da rede e as aplicações. Esta comunicação com os *switches* é realizada através do uso da biblioteca Libfluid [22], utilizada para prover aos controladores SDN uma interface, ou seja, fornecer comandos básicos para comunicação entre os controladores e os equipamentos da rede.

A arquitetura proposta consiste na apresentada na Figura 5. Esta arquitetura corresponde a do RunOS com a adição do BomIP, que é consumida na própria classe do controlador do RunOS no momento em que um pacote é processado para ser enviado na rede.

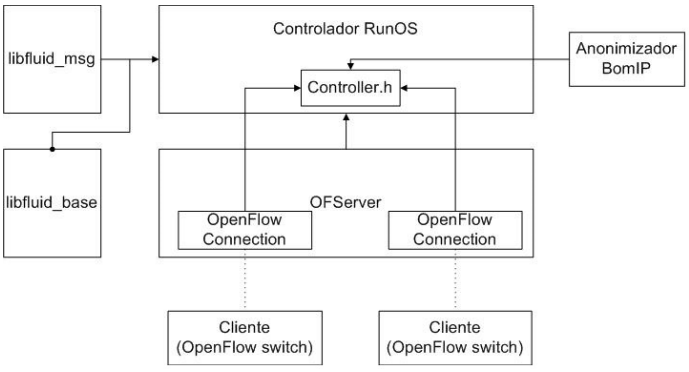


Figura 5. Arquitetura Proposta de um Serviço de Anonimização em SDN

No funcionamento da arquitetura, todo pacote tem seus endereços IP de origem e destino analisados no controlador através do anonimizador BomIP. Na tabela de endereços IP da rede que fica armazenada no controlador para indicar as rotas que os pacotes devem seguir, os endereços são armazenados anonimizados. Ou seja, quando um novo endereço é adicionado, o mesmo é anonimizado utilizando o BomIP e armazenado.

O algoritmo do BomIP na etapa de anonimização, no pior caso, analisa cada pacote para processar os endereços IP de origem e destino. Porém, é necessário validar se o valor anonimizado já não existe, e com isto percorrer o vetor, o que, no pior caso, pode ter que percorrer as n posições existentes. Segundo [23], os processos de atribuição de valor, atribuir valor aleatório e acesso direto a posição, são etapas de tempo constante, porém, percorrer um vetor de tamanho n, no pior caso, custa tempo n. Assim, a análise assintótica do algorítmo é expressa pelo seguinte somatório $\sum_{i=1}^{n} i$, com o resultado da expansão deste somatório sendo o próprio n^2, conforme a expansão $\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$. Logo, o algoritmo é executado em tempo O(n^2).

V. Trabalhos Relacionados

Com o objetivo de realizar uma randomização para aumentar a detecção de ataques enquanto os mesmos ocorrem, em [24] é introduzido o conceito de "Defesa Proativa contra Adversário Consciente". Esse mecanismo tem como alvo estabelecer, estrategicamente, um dinamismo em sistemas estáticos para caracterizar o comportamento do adversário. A abordagem realiza a anonimização do endereço IP das estações frequentemente, através da geração de novos identificadores aleatórios, de forma a não permitir a rastreabilidade do reconhecimento da rede para ataques. A distribuição baseada em quais endereços são designados para cada estação da rede, bem como a taxa de randomização é adaptativa e determinada pela verificação do endereço que está sob ataque.

Para realizar a reconfiguração de uma rede em tempo real, Chavez, Stout e Peisert (2015) propõem o sistema "Moving Target Defense" (MTD), ou Sistema de Alvo em Movimento, que garante a não interrupção da conectividade entre os nós da rede. O sistema MTD consiste de três técnicas: a randomização de portas TCP/UDP, a randomização de endereços IP, e a randomização dos caminhos de uma rede. A randomização de endereços IP ainda permite que a rede esteja suscetível a análise de tráfego através de ferramentas do tipo *sniffer*. Para impedir o monitoramento da comunicação dos pacotes trafegados na rede, os autores propõem a randomização dos caminhos. Para cada fluxo da rede, o controlador designa um caminho randomizado entre os nós da rede que estão se comunicando.

O trabalho de [26] concentra-se no campo referente a variação do endereço IP, de forma a modificá-lo antes que ele entre em uma rede pública, prevenindo que o mesmo possa ser identificado. Para esta finalidade, os autores desenvolvem três módulos para realizar a randomização de endereços IP, gerenciados pelo controlador SDN. Os módulos consistem em um módulo de conversão entre endereços IP virtuais e reais, um módulo de autenticação e um gerador de endereços IP virtuais. Os resultados obtidos com experimentos indicaram que o trabalho de [26] possui uma taxa de 95% de eficiência, impedindo assim, que a maioria dos endereços IP reais fossem

descobertos durante as tentativas simuladas de descoberta de endereços.

AnonyFlow [3] é uma solução em que usuários da Internet utilizam um endereço de IP temporário e descartado por fluxo. Esta abordagem é similar ao conceito de *Network Address Translation* (NAT): terceiros não tem visibilidade sobre a rede interna, e assim, não conseguem correlacionar o tráfego com um endereço IP. Os autores apresentam essa característica como um serviço adicional que pode ser escolhido, similar ao ocultamento de números em chamadas telefônicas.

OF-RHM (OpenFlow Random Host Mutation) [27] é um mecanismo para realizar "mutação" no endereço IP de uma rede randomicamente e frequentemente de forma transparente. Neste caso, o controlador designa cada estação um IP virtual temporário que é traduzido "de" e "para" em relação aos endereços IP reais das estações. Os resultados teóricos indicaram que OF-RHM invalidou 99% da tentativa de obtenção de informação através de escaneamento remoto.

Nestes trabalhos, a anonimização dos identificadores ocorre de forma randômica, o que não permite que uma análise de tráfego seja realizada na rede, uma vez que os identificadores não mantém características como a sub-máscara da rede. Por isto, é possível visualizar também na Tabela I a lacuna em relação a permitir uma análise do tráfego após a anonimização ter sido realizada na rede.

Tabela I
COMPARAÇÃO DOS TRABALHOS RELACIONADOS

Autores	**Anonimização**	**IP**	**Rastreabilidade do Tráfego**	**Novo IP por Fluxo**
[24]	Aleatória	X		
[25]	Aleatória	X		
[26]	Aleatória	X		
[3]	Aleatória	X		X
[27]	Aleatória	X		X
Serviço BomIP	Aleatória com preservação de prefixo	X	X	

Diferente de outros modelos propostos, o BomIP oferece anonimização em tempo real dos endereços IP dos pacotes trafegados na rede, para oferecer assim, um serviço de segurança provendo privacidade dos *hosts* e outros serviços. Atualmente, um intruso na rede pode verificar um comportamento de um endereço IP para descobrir os serviços existentes na rede e, assim, obter portas para realizar ataques. Desta forma, o BomIP é projetado para esconder as informações de endereços da rede de um invasor, para impedir que este usuário possa causar algum dano aos serviços ou dados trafegados.

VI. ESTUDOS DE CASO

Para realizar os Estudos de Caso I e II foram utilizados os traces disponibilizados pelo *Lincoln Laboratory do Massachusetts Institute of Technology* (MIT). Estes traces foram criados em laboratório pelo grupo de pesquisa *Intrusion Detection Evaluation Group*, denominado DARPA [28].

O Estudo de Caso III demonstra a mitigação dos ataques de Negação de Serviços sofridos por uma SDN, em um ambiente simulado. Os resultados deste ataque são analisados com uma mesma rede sem o uso de serviço de anonimização, e desta forma, é possível comparar o percentual de estações atacadas em cada uma das redes durante um período de tempo, sendo possível verificar a mitigação em relação à rede com uso do serviço de anonimização.

A. Estudo de Caso I

O primeiro Estudo de Caso consiste em comparar o tempo de execução dos algoritmos de anonimização do BomIP e do Crypto-Pan [18]. Esta comparação é realizada através de análise assintótica dos algoritmos, considerando o valor já calculado do BomIP de O(n^2).

Para realizar a anonimização, o Crypto-Pan utiliza o algoritmo de cifra de Rijndael [29]. Este corresponde a algoritmo de cifra de chave simétrica de 128 ou 256 bits, em que a única forma conhecida para quebrar o segredo é através de algoritmos de força bruta [30], o que garante uma maior segurança e confiabilidade a uma informação anonimizada pelo Crypto-Pan.

A partir da análise realizada por [31] é constatado que o algoritmo de Rijdael, utilizado no Crypto-Pan, tem uma complexidade algorítmica de O(n^4), o que é possível verificar devido a presença de dois laços de repetição aninhados, em que é chamada a função de criptografia. Esta função de criptografia faz um caminhamento em uma matriz.

Utilizando a base de dados do MIT, com os resultados da segunda semana de simulação do ano de 1999, os endereços IP dos pacotes capturados nesta simulação foram anonimizados em ambas as ferramentas, BomIP e Crypto-Pan, de forma a validar os resultados de complexidade obtidos através de um processamento real.

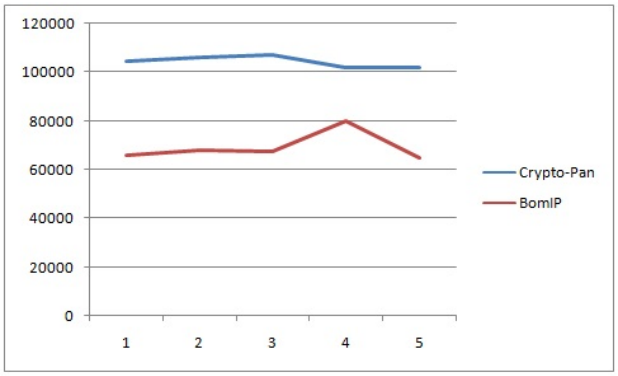
Os resultados obtidos no processamento estão apresentados nas Figuras 6(a) e 6(b), em que a linha superior em cada gráfico representa o Crypto-pan, enquanto que a linha inferior corresponde ao tempo do BomIP.

Com base na análise dos gráficos, é possível verificar que o ganho de processamento do BomIP em relação ao Crypto-Pan corresponde a 65,5% (segunda-feira: 66,27% e terça-feira: 64,56%), conforme também pode-se observar através da análise assintótica explicada anteriormente.

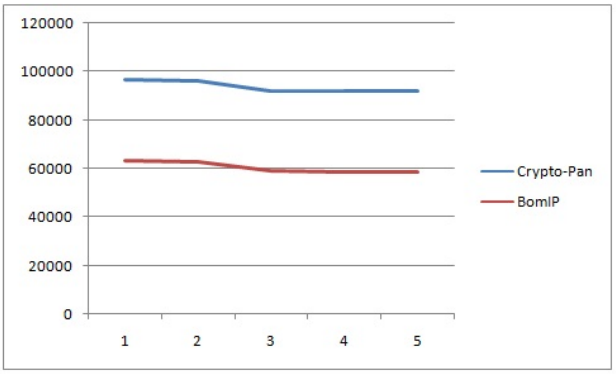
B. Estudo de Caso II

O segundo Estudo de Caso, baseado no trabalho de [32], consiste em anonimizar o *trace* original, e realizar a análise na ferramenta SNORT, e verificar desta forma a rastreabilidade dos traces.

A Figura 7 apresenta o fluxograma do experimento realizado. Inicialmente foram obtidos os traces correspondentes a segunda semana de medição disponibilizada pelo grupo DARPA [28] do MIT. Em seguida, estes traces originais foram anonimizados pelo Crypto-Pan, gerando um novo conjunto de traces; e também os traces originais foram anonimizados com o BomIP, ocasionando um novo conjunto de traces anonimizados.



(a) Segunda-Feira



(b) Terça-Feira

Figura 6. Tempo de processamento em milissegundos de amostras na segunda-feira (a) e terça-feira (b)

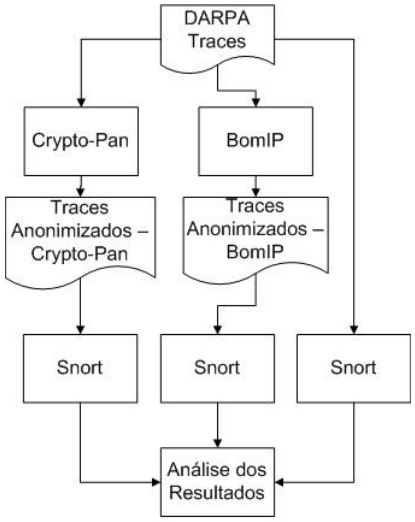


Figura 7. Fluxograma do experimento realizado.

Os resultados estão apresentados nas Tabelas II, III, IV, V e VI, contendo os valores obtidos a partir da análise de cada um dos LOGs gerados pelo SNORT para cada trace original e anonimizado.

Estes resultados apontam uma eficiência em rastreabilidade do BomIP semelhante ao Crypto-Pan, o que pode ser comprovado pela quantidade de conexões TCP, que são mantidas em relação ao número do *trace* original. Além da quantidade de pacotes anonimizados, e de Falso-Positivos e Falso-Negativos encontrados em cada um dos *traces* anonimizados pelas ferramentas, que demonstram que os algoritmos de anonimização dos pacotes possuem um percentual próximo de coesão.

Tabela II
DADOS ESTATÍSTICOS REFERENTES AOS *traces* DARPA DE SEGUNDA-FEIRA

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.753.377	1.738.745	1.738.745
Positivos	64.651	64.649	64.649
Falso-Positivos	-	2.329	2.330
Falso-Negativos	-	2	2
Nº Pacotes Anonimizados	-	1.738.745	1.738.745
Nº conexões TCP	44.491	44.491	44.491

Tabela III
DADOS ESTATÍSTICOS REFERENTES AOS *traces* DARPA DE TERÇA-FEIRA

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.585.120	1.572.881	1.572.881
Positivos	52.903	52.903	52.903
Falso-Positivos	-	2.503	2.508
Falso-Negativos	-	0	0
Nº Pacotes Anonimizados	-	1.572.881	1.572.881
Nº conexões TCP	55.503	55.503	55.503

Tabela IV
DADOS ESTATÍSTICOS REFERENTES AOS *traces* DARPA DE QUARTA-FEIRA

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.011.149	996.547	996.547
Positivos	32.347	32.347	32.347
Falso-Positivos	-	1.732	1.750
Falso-Negativos	-	0	0
Nº Pacotes Anonimizados	-	996.547	996.547
Nº conexões TCP	26.219	26.219	26.219

Tabela V
DADOS ESTATÍSTICOS REFERENTES AOS *traces* DARPA DE QUINTA-FEIRA

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.563.069	1.548.993	1.548.993
Positivos	48.719	48.715	48.715
Falso-Positivos	-	1.977	1.962
Falso-Negativos	-	4	4
Nº Pacotes Anonimizados	-	1.548.993	1.548.993
Nº conexões TCP	68.121	68.121	68.121

Tabela VI
DADOS ESTATÍSTICOS REFERENTES AOS *traces* DARPA DE SEXTA-FEIRA

	Original	*Crypto-Pan*	*BomIP*
Nº de Pacotes	1.362.422	1.348.705	1.348.705
Positivos	57.987	57.987	57.987
Falso-Positivos	-	1.534	1.534
Falso-Negativos	-	0	0
Nº Pacotes Anonimizados	-	1.348.705	1.348.705

C. Estudo de Caso III

O terceiro Estudo de Caso, baseado nos trabalhos de [33] e [26], é realizado através de uma rede simulada, criada

utilizando um *script* desenvolvido em *Python* para *Mininet*.

O objetivo deste experimento é apresentar a mitigação do quantitativo de ataques que a rede sofre quando os endereços IP contidos na Tabela de IP do controlador encontra-se anonimizado. Esta rede possui o roteador principal com um endereço IP de uma máscara de 20 bits e duas sub-redes, conforme apresentado na Figura 8.

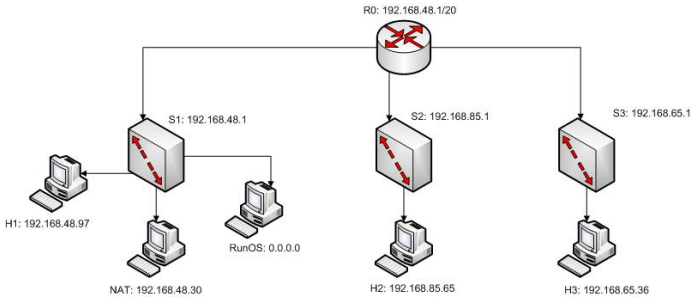


Figura 8. Ambiente simulado criado no Mininet

Iniciou-se um ataque à rede tomando como ponto de partida que o usuário conseguiu acesso à um *host*. Com base neste *host*, o usuário conseguiu acesso a Tabela de IPs do controlador RunOS. Quando realizou o acesso ao host, o invasor consegue inferir:

1) Os valores anonimizados 199 e 160 correspondem, respectivamente, a 192 e 168 não anonimizados. Isto porque o IP utilizado para ataque corresponde a 192.168.48.30.
2) Existem 3 sub-redes, 192.168.23.X, 192.168.25.Y e 192.168.73.Z, em que 23, 25 e 73 são valores anonimizados de forma randômica, e que um destes valores corresponde a 48.
3) Os *hosts* disponíveis correspondem ao último octeto, todos anonimizados, com valores 8, 13, 23, 48, 89, e um destes valores corresponde a 30.

Em cada tentativa de localização de um *host*, é usado o programa NMAP para verificar a existência do *host* e se há porta aberta disponível para ataque. Caso tenha sucesso, é usado o programa DDOSIM [34] para realizar um ataque de DDoS no *host* localizado na porta aberta indicada pelo NMAP.

De acordo com [35], algoritmos de criptografia ou randômicos são quebrados através de técnicas de força bruta, ou seja, o usuário (invasor) deve tentar todas as possibilidades disponíveis. Isto implica que, a partir do momento que não há uma dessas técnicas impostas, o invasor pode tentar um acesso direto ou usar algoritmos para conseguir obter com mais facilidade os valores possíveis. Assim, pode-se ter dois sub-cenários.

1) A rede não se encontra anonimizada pelo controlador. Logo, quando o invasor entrar na rede, conseguirá todos os endereços IP reais, e assim, pode realizar um ataque de DDoS de forma direta em cada uma das estações desejadas.
2) A rede encontra-se anonimizada. Assim, o usuário tem apenas o conhecimento dos octetos do host que conseguiu realizar a invasão, e com isso, apenas uma das 3 classes de IP como conhecimento, tendo que usar todos os bits possíveis de hosts da classe para tentar atacar a rede.

Considerando a invasão, foi realizada uma tentativa de ataque em cada um dos sub-cenários, e foi aferido um tempo médio de 2 segundos por IP tentado a ser atacado. Desta forma, a Figura 9 apresenta o resultado obtido. O eixo X corresponde ao tempo decorrido de ataque, enquanto que na linha do gráfico (eixo Y) é possível verificar o percentual de *hosts*.

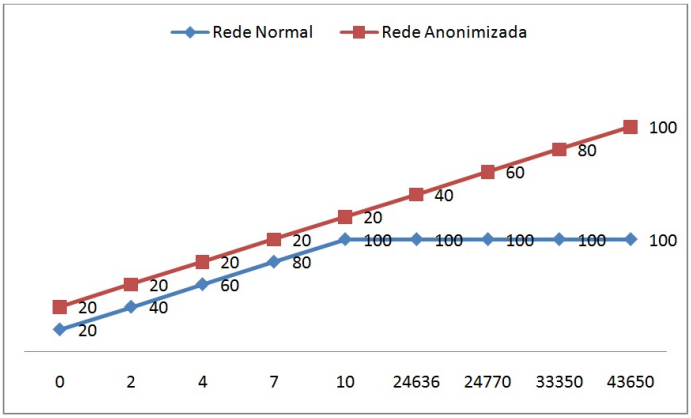


Figura 9. Arquitetura de tentativa de ataque à rede

Importante visualizar que quando a rede normal teve a totalidade de seus *hosts* sob ataque, a rede anonimizada tinha 20% afetada. Esta diferença no tempo mostra a mitigação do ataque a que a rede anonimizada está sofrendo, bem como permite que medidas sejam tomadas quando um ataque é iniciado, já que há um intervalo entre o ataque em uma estação e outra.

Assim, no pior cenário, o invasor pode ter que percorrer todo o intervalo possível da faixa de IP de *hosts*, com base na quantidade de bits disponíveis na máscara da classe do IP. Neste exemplo, considerando um tempo gasto de 2 segundos por tentativa de invasão a um *host*, e sabendo que uma máscara de classe B possui até 65536 endereços, o que levaria um tempo de, aproximadamente, 36 horas de tentativa de ataque.

VII. Conclusões e Trabalhos Futuros

Este trabalho apresentou uma solução voltada para segurança de Redes Definidas por Software através de um serviço de anonimização de redes SDN sem causar inconsistência. Diante dos resultados apresentados, foi possível perceber que a utilização do serviço proposto, apresentou-se de forma positiva, com resultados que mitigaram a quantidade de ataques sofridos pela rede em comparação a uma rede que não utilizava o serviço de anonimização, conduzindo a um maior tempo até que a rede fosse invadida, permitindo assim, que administradores da rede pudessem tomar medidas para conter a invasão.

Como trabalhos futuros, almeja-se a extensão do serviço de anonimização para outros campos, como endereços MAC e

portas, garantindo mais segurança a rede. Além também de permitir que pacotes IPV6 também possam ser anonimizados.

REFERÊNCIAS

[1] A. Akhunzada, E. Ahmed, A. Gani, M. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Communications Magazine*, vol. 54, pp. 36–44, 2015.

[2] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: pros and cons," *IEEE Communications Magazine*, vol. 53, pp. 73–79, 2015.

[3] M. Mendonca, S. Seetharaman, and K. Obraczka, "A flexible in-network ip anonymization service," *IEEE Int. Conf. Communications*, vol. 1, pp. 6651–6656, 2012.

[4] D. Koukis, S. Antonatos, D. Antoniades, E. Markatos, and P. Trimintzios, "A generic anonymization framework for network traffic," *IEEE International Conference on Communications*, vol. 1, pp. 2302–2309, 2006.

[5] S. Mahajan, A. M. Adagale, and C. Sahare, "Intrusion detection system using raspberry pi honeypot in network security," *International Journal of Engineering Science*, vol. 2792, 2016.

[6] S. Scott-Hayward, "Design and deployment of secure, robust, and resilient sdn controllers," *IEEE Conference on Network Softwarization*, vol. 1, pp. 01–05, 2015.

[7] A. Feghali, R. Kilany, and M. Chamoun, "Sdn security problems and solutions analysis," *International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, vol. 1, pp. 01–05, 2015.

[8] D. K. F. R. P. E. V. C. E. R. S. A. S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 130, pp. 14–76, 2015.

[9] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, 2014.

[10] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi, D. Meyer, and O. Koufopavlou, "Software-defined networking (sdn): Layers and architecture terminology," 1 2015, rFC 7426.

[11] O. N. Foundation, "Openflow switch specification v1.3.0," https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflowspec-v1.3.0.pdf, 2012, accessed: 2015-06-15.

[12] V. Paulo, "Towards secure and dependable software-defined networks," *ACM SIGCOMM HotSDN*, vol. 1, pp. 55–60, 2013.

[13] Y. Sung, P. K. Sharma, E. M. Lopez, and J. H. Park, "Fs-opensecurity: A taxonomic modeling of security threats in sdn for future sustainable computing," *Sustainability*, vol. 8, no. 9, p. 919, 2016.

[14] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," *IEEE 35th Conference on Local Computer Networks*, vol. 1, pp. 408–415, 2010.

[15] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using opensafe," *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, vol. 1, 2010.

[16] T. Farah and L. Trajkovi, "Anonym: A tool for anonymization of the internet traffic," *IEEE International Conference on Cybernetics (CYBCONF)*, vol. 1, pp. 261–266, 2013.

[17] S. Dara, "Network telemetry anonymization for cloud based," *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, vol. 1, pp. 01–07, 2014.

[18] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving ip address anonymization: measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253–272, 2004.

[19] L. M. Garcia, "Programming with libpcap-sniffing the network from our own application," *Hakin9-Computer Security Magazine*, pp. 2–2008, 2008.

[20] V. Jacobson and S. McCanne, "libpcap: Packet capture library," *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009.

[21] A. Shalimov, S. Nizovtsev, D. Morkovnik, and R. Smeliansky, "The runos openflow controller," in *2015 Fourth European Workshop on Software Defined Networks*. IEEE, 2015, pp. 103–104.

[22] A. Vidal, C. E. Rothenberg, and F. L. Verdi, "The libfluid openflow driver implementation," in *32nd Brazilian Symposium on Computer Networks (SBRC). SBC*, 2014, p. 8.

[23] U. Manber, *Introduction to algorithms: a creative approach*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[24] H. Jafarian and E. Al-Shaer, "Adversary-aware ip address randomization for proactive agility against sophisticated attackers," *IEEE Conference on Computer Communications*, vol. 1, pp. 738–746, 2015.

[25] A. Chavez, W. Stout, and S. Peisert, "Techniques for the dynamic randomization of network attributes," *Proceedings of the 49th Annual International Carnahan Conference on Security Technology*, vol. 1, pp. 01–06, 2015.

[26] Z. Zhao, Y. Guo, and W. Liu, "The design and research for network address space randomization in openflow network," *Journal of Computer and Communications*, vol. 3, pp. 203–211, 2015.

[27] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," *ACM Workshop Hot Topics in Software Defined Networks*, vol. 1, pp. 127–132, 2012.

[28] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer networks*, vol. 34, no. 4, pp. 579–595, 2000.

[29] J. Daemen and V. Rijmen, "The block cipher rijndael," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 1998, pp. 277–284.

[30] H. K. Verma and R. K. Singh, "Performance analysis of rc6, twofish and rijndael block cipher algorithms," *International Journal of Computer Applications (0975–8887) Volume*, pp. 1–7, 2012.

[31] C. I. S. Uy, J. A. L. Angeles, N. A. Flores, A. K. D. Balan, and J. F. Abisado, "Flaxor: A symmetric-key block cipher plug-in using usb mass storage device for extracted block storage," *JUPITER: 1st ITE Research Colloquium/TIP-QC/ March 28*, vol. 1, 2008.

[32] K. Lakkaraju and A. Slagell, "Evaluating the utility of anonymized network traces for intrusion detection," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*. ACM, 2008, p. 17.

[33] N. G. Dharma, M. F. Muthohar, J. A. Prayuda, K. Priagung, and D. Choi, "Time-based ddos detection and mitigation for sdn controller," in *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. IEEE, 2015, pp. 550–553.

[34] C. Ye and K. Zheng, "Detection of application layer distributed denial of service," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 1. IEEE, 2011, pp. 310–314.

[35] T. Brekne, A. Årnes, and A. Øslebø, "Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies," in *International Workshop on Privacy Enhancing Technologies*. Springer, 2005, pp. 179–196.